PASSNYC MAIN

Manoranjan Kumar

7/23/2018

# Read the saved scv back again  
SchoolMain <- read.csv("https://raw.githubusercontent.com/manusatx/NYC\_SCHOOLS/master/SchoolMain.csv")  
dim(SchoolMain)  
## [1] 1272 52

# data exploration for predictors Remove ID and name and other Identificable  
# varaibles by looking at the data - 27 variables removed  
SchoolMain <- subset(SchoolMain, select = -c(X, District, New., Adjusted.Grade,   
 GEOID, Other.Location.Code.in.LCGMS, SchoolGEOID, School.Name, SED.Code,   
 Location.Code, Latitude, Longitude, Address..Full., City, Zip, Grades, Grade.Low,   
 Grade.High, Rigorous.Instruction.Rating, Collaborative.Teachers.Rating,   
 Supportive.Environment.Rating, Effective.School.Leadership.Rating, Strong.Family.Community.Ties.Rating,   
 Trust.Rating, Student.Achievement.Rating))  
dim(SchoolMain)  
## [1] 1272 27

# Convert community to 1 / 0 vs Yes No  
SchoolMain$Community.School. <- ifelse(SchoolMain$Community.School. == "Yes",   
 1, 0)  
  
# Convert all columns to numeric  
SchoolMain <- as.data.frame(lapply(SchoolMain, function(x) as.numeric(as.character(x))))  
## Warning in FUN(X[[i]], ...): NAs introduced by coercion  
  
## Warning in FUN(X[[i]], ...): NAs introduced by coercion  
  
## Warning in FUN(X[[i]], ...): NAs introduced by coercion  
  
## Warning in FUN(X[[i]], ...): NAs introduced by coercion  
  
## Warning in FUN(X[[i]], ...): NAs introduced by coercion  
  
## Warning in FUN(X[[i]], ...): NAs introduced by coercion  
  
## Warning in FUN(X[[i]], ...): NAs introduced by coercion  
  
## Warning in FUN(X[[i]], ...): NAs introduced by coercion  
  
## Warning in FUN(X[[i]], ...): NAs introduced by coercion  
  
## Warning in FUN(X[[i]], ...): NAs introduced by coercion  
  
## Warning in FUN(X[[i]], ...): NAs introduced by coercion  
  
## Warning in FUN(X[[i]], ...): NAs introduced by coercion

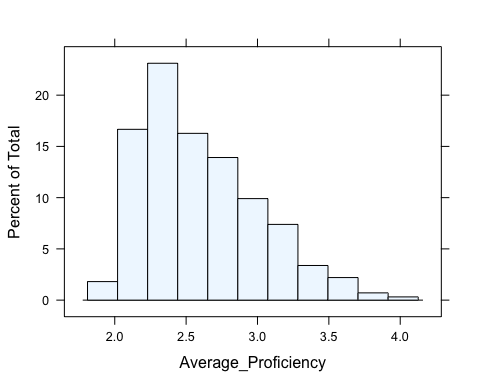
# Make Community.School as factor  
SchoolMain$Community.School. <- as.factor(SchoolMain$Community.School.)  
  
# Merge the ELA and maths score to form one response variable column  
SchoolMain$Average\_Proficiency <- (SchoolMain$Average.ELA.Proficiency + SchoolMain$Average.Math.Proficiency)/2  
SchoolMain <- subset(SchoolMain, select = -c(Average.ELA.Proficiency, Average.Math.Proficiency))  
  
# Missing values  
summary(SchoolMain)  
## Community.School. Economic.Need.Index School.Income.Estimate  
## 0:1196 Min. :0.0500 Min. : 16902   
## 1: 76 1st Qu.:0.5500 1st Qu.: 33610   
## Median :0.7300 Median : 43151   
## Mean :0.6728 Mean : 48443   
## 3rd Qu.:0.8400 3rd Qu.: 58518   
## Max. :0.9600 Max. :181382   
## NA's :25 NA's :396   
## Percent.ELL Percent.Asian Percent.Black Percent.Hispanic  
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0200   
## 1st Qu.:0.0400 1st Qu.:0.0100 1st Qu.:0.0600 1st Qu.:0.1800   
## Median :0.0900 Median :0.0400 Median :0.2400 Median :0.3550   
## Mean :0.1248 Mean :0.1165 Mean :0.3200 Mean :0.4115   
## 3rd Qu.:0.1700 3rd Qu.:0.1400 3rd Qu.:0.5525 3rd Qu.:0.6400   
## Max. :0.9900 Max. :0.9500 Max. :0.9700 Max. :1.0000   
##   
## Percent.Black...Hispanic Percent.White Student.Attendance.Rate  
## Min. :0.0300 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.4900 1st Qu.:0.0100 1st Qu.:0.9200   
## Median :0.9000 Median :0.0300 Median :0.9400   
## Mean :0.7314 Mean :0.1316 Mean :0.9272   
## 3rd Qu.:0.9600 3rd Qu.:0.1600 3rd Qu.:0.9500   
## Max. :1.0000 Max. :0.9200 Max. :1.0000   
## NA's :25   
## Percent.of.Students.Chronically.Absent Rigorous.Instruction..  
## Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.1100 1st Qu.:0.8600   
## Median :0.2000 Median :0.9000   
## Mean :0.2157 Mean :0.8948   
## 3rd Qu.:0.3000 3rd Qu.:0.9400   
## Max. :1.0000 Max. :1.0000   
## NA's :25 NA's :25   
## Collaborative.Teachers.. Supportive.Environment..  
## Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.8500 1st Qu.:0.8400   
## Median :0.9000 Median :0.8900   
## Mean :0.8844 Mean :0.8875   
## 3rd Qu.:0.9400 3rd Qu.:0.9400   
## Max. :1.0000 Max. :1.0000   
## NA's :25 NA's :25   
## Effective.School.Leadership.. Strong.Family.Community.Ties..  
## Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.7600 1st Qu.:0.8000   
## Median :0.8300 Median :0.8300   
## Mean :0.8162 Mean :0.8309   
## 3rd Qu.:0.8900 3rd Qu.:0.8700   
## Max. :0.9900 Max. :0.9900   
## NA's :25 NA's :25   
## Trust.. NumOfLibraries NumOfAfterSchoolProgs  
## Min. :0.0000 Min. : 0.000 Min. : 0.00   
## 1st Qu.:0.8700 1st Qu.: 2.000 1st Qu.:13.00   
## Median :0.9200 Median : 3.000 Median :25.00   
## Mean :0.9042 Mean : 3.169 Mean :29.77   
## 3rd Qu.:0.9400 3rd Qu.: 4.000 3rd Qu.:47.00   
## Max. :1.0000 Max. :10.000 Max. :80.00   
## NA's :25   
## NumOfLowIncomeUnits MedianHldIncome IncomeToPovertyRatio  
## Min. : 0.00 Min. : 12052 Min. : 0   
## 1st Qu.: 1.00 1st Qu.: 31492 1st Qu.: 658   
## Median : 15.00 Median : 47518 Median : 930   
## Mean : 34.41 Mean : 53655 Mean :1037   
## 3rd Qu.: 62.00 3rd Qu.: 68854 3rd Qu.:1314   
## Max. :141.00 Max. :250001 Max. :7004   
## NA's :8   
## ChildPoverty CountofSNAPHlds HealthCoverage Average\_Proficiency  
## Min. : 0.0 Min. : 0.0 Min. : 0 Min. :1.895   
## 1st Qu.: 263.0 1st Qu.: 593.2 1st Qu.: 2934 1st Qu.:2.275   
## Median : 751.0 Median : 1339.0 Median : 4212 Median :2.515   
## Mean : 830.6 Mean : 1473.0 Mean : 4665 Mean :2.602   
## 3rd Qu.:1243.0 3rd Qu.: 2000.0 3rd Qu.: 5964 3rd Qu.:2.860   
## Max. :5420.0 Max. :12683.0 Max. :28186 Max. :4.040   
## NA's :55

# check only for response variable Yield  
nrow(SchoolMain[is.na(SchoolMain$Average\_Proficiency), ])  
## [1] 55

print("There are 55 rows with some missing data")  
## [1] "There are 55 rows with some missing data"

# Check Distribution of response variable  
library(AppliedPredictiveModeling)  
## Warning: package 'AppliedPredictiveModeling' was built under R version  
## 3.4.4

library(lattice)  
transparentTheme(pchSize = 1, trans = 0.7)  
mean.values = mean(SchoolMain$Average\_Proficiency)  
histogram(~Average\_Proficiency, data = SchoolMain, xlab = "Average\_Proficiency",   
 panel = function(x, ...) {  
 panel.histogram(x, ...)  
 panel.abline(v = mean.values, col.line = "red", identifier = "abline",   
 lty = 4)  
 })



# Remove 55 rows from data where response variable is NULL  
SchoolMain\_NA\_Resp <- SchoolMain[is.na(SchoolMain$Average\_Proficiency), ]  
SchoolMain <- SchoolMain[!is.na(SchoolMain$Average\_Proficiency), ]  
  
# see predictors data with missing values  
nrow(SchoolMain[rowSums(is.na(subset(SchoolMain, select = -Average\_Proficiency))) >   
 0, ])  
## [1] 394

print("There are 394 rows with some missing data")  
## [1] "There are 394 rows with some missing data"

# Use Knn imputation to preprocess the data  
library(caret)  
SchoolMain\_prepro\_index <- preProcess(subset(SchoolMain, select = -Average\_Proficiency),   
 method = c("BoxCox", "knnImpute"))  
SchoolMain\_new <- predict(SchoolMain\_prepro\_index, SchoolMain)  
  
# see predictors data with missing values after imputation  
nrow(SchoolMain\_new[rowSums(is.na(subset(SchoolMain\_new, select = -Average\_Proficiency))) >   
 0, ])  
## [1] 0

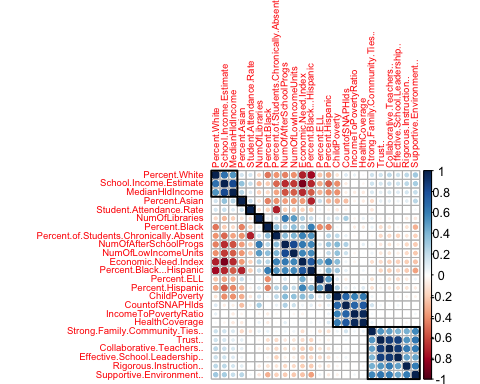
print("There are 0 rows with some missing data after knn imputation")  
## [1] "There are 0 rows with some missing data after knn imputation"

# Create a df with predictors only and without the factor columns has 24  
# columns  
SchoolMain\_new\_numeric <- SchoolMain\_new[, sapply(SchoolMain\_new[, !names(SchoolMain\_new) %in%   
 c("Average\_Proficiency")], is.numeric)]  
  
# df with factor and response has 2 columns  
SchoolMain\_fctr\_resp <- subset(SchoolMain\_new, select = c(Community.School.,   
 Average\_Proficiency))

# check for near zero variance variables  
library(caret)  
rm\_SchoolMain\_cols <- nearZeroVar(SchoolMain\_new)  
rm\_SchoolMain\_cols  
## integer(0)

print("there is no columns with near zero variance")  
## [1] "there is no columns with near zero variance"

# Check the data numeric volumns for correlation for 24 predictors  
library(caret)  
SchoolMaincorr = cor(SchoolMain\_new\_numeric)  
corrplot::corrplot(SchoolMaincorr, order = "hclust", addrect = 9, method = "circle",   
 tl.cex = 0.6)

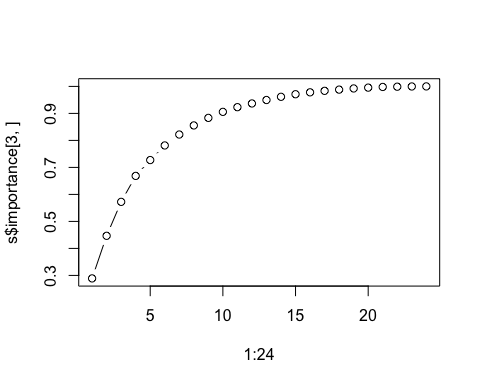


# library(ggcorrplot) ggcorrplot(SchoolMaincorr ,hc.order = TRUE, method =  
# c('square'),tl.cex = 7, show.diag = FALSE, ggtheme =  
# ggplot2::theme\_minimal)  
  
# Check for correlated varaibles and remove highly correlated predictors  
SchoolMaincorrpp <- findCorrelation(SchoolMaincorr, cutoff = 0.8)  
SchoolMain\_no\_cor <- SchoolMain\_new\_numeric[, -SchoolMaincorrpp]  
dim(SchoolMain\_new\_numeric)  
## [1] 1217 24

dim(SchoolMain\_no\_cor)  
## [1] 1217 18

print("From the continuous variables 7 correlated predictor variables were removed with correlation check and now we have 24-7 = 17 predictor variables")  
## [1] "From the continuous variables 7 correlated predictor variables were removed with correlation check and now we have 24-7 = 17 predictor variables"

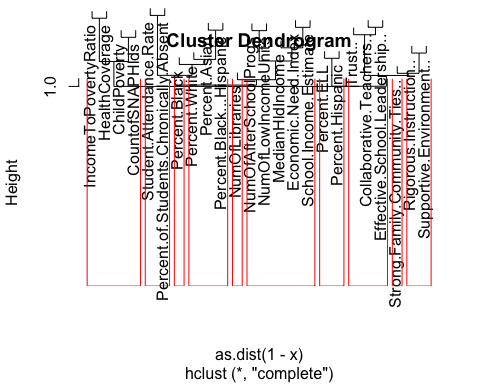
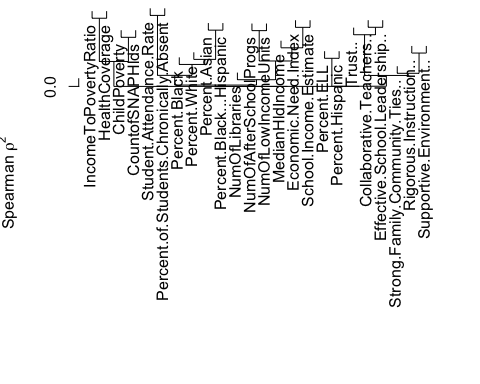
# PCA  
SchoolMain.pca <- prcomp(SchoolMain\_new\_numeric, center = TRUE, scale. = TRUE)  
s <- summary(SchoolMain.pca)  
# Plot cumulative variance Principal components  
plot(1:24, s$importance[3, ], type = "b")



print("PC1, PC2 till PC10 explain 90% of variance cumulatively as seen from the summary")  
## [1] "PC1, PC2 till PC10 explain 90% of variance cumulatively as seen from the summary"

# Plot the resultant Principal components biplot(SchoolMain.pca, scale = 1,  
# cex = 0.6)  
  
# with removed highly correlated variables datset is now having 19 columns  
# and 1217 rows  
SchoolMain\_reducedcorr <- cbind(SchoolMain\_no\_cor, SchoolMain\_fctr\_resp)  
dim(SchoolMain\_reducedcorr)  
## [1] 1217 20

# variable clustering and dimesnionality reduction calculate R2 Ratio for  
# each variable in each cluster. We will then select variable with minimum  
# 1-R2 ratio in each cluster as cluster representative  
r2clusterfun <- function(inputdf, insim, numcuttree) {  
 library(Hmisc)  
 varclus.inputdf <- varclus(data.matrix(inputdf), similarity = insim)  
 plot(varclus.inputdf)  
 # Below is hclust object  
 hclust.inputdf <- varclus.inputdf$hclust  
 # plot the 10 clusters by cut  
 plot(hclust.inputdf)  
 rect.hclust(hclust.inputdf, k = numcuttree, border = "red")  
 # Cut the above into 10 clusters and assign name to data frame  
 groups.inputdf <- as.data.frame(cutree(hclust.inputdf, k = numcuttree))  
 library(data.table)  
 groups.inputdf <- as.data.frame(setDT(groups.inputdf, keep.rownames = TRUE)[])  
 colnames(groups.inputdf) <- c("var", "clusternum")  
 groups.inputdf$ID <- seq.int(nrow(groups.inputdf))  
 groups.inputdf$ID <- as.character(groups.inputdf$ID)  
 groups.inputdf.index.list <- as.list(as.matrix(by(groups.inputdf$ID, groups.inputdf$clusternum,   
 function(x) return(as.numeric(as.character(x))))))  
 # Check for index lists > 1  
 index.list <- groups.inputdf.index.list[sapply(groups.inputdf.index.list,   
 length) > 1]  
 index.list.1 <- groups.inputdf.index.list[sapply(groups.inputdf.index.list,   
 length) == 1]  
 # Disimilarity matrix (1-r^2) using Pearson  
 cor.inputdf <- cor(inputdf, method = insim)  
 cormatrix <- round((1 - (cor.inputdf)^2), 3)  
 # cormatrix is dissimilarity matrix Dissimalrity (1-r^2) between elements of  
 # a cluster and the other elements in its own cluster  
 h <- function(index) {  
 temp <- cormatrix[index, index]  
 diag(temp) <- NA  
 apply(temp, 1, min, na.rm = T)  
 }  
 numer <- lapply(index.list, h)  
 # Dissimalrity (1-r^2) between elements of each cluster and other clusters  
 g <- function(index) {  
 apply(cormatrix[-index, index], 2, min)  
 }  
 denom <- lapply(index.list, g)  
 # Find the minimum r^2 ratio  
 i <- function(index) {  
 which.min(numer[[index]]/denom[[index]])  
 }  
 apply(as.matrix(1:length(index.list)), 1, i)  
 # get the index of each cluster lowest r2 element  
 min\_r2\_index <- as.data.frame(apply(as.matrix(1:length(index.list)), 1,   
 i))  
 colnames(min\_r2\_index) <- c("min\_r2\_idx")  
 min\_r2\_index$id <- seq.int(nrow(min\_r2\_index))  
   
 print("The 10 variables with the lowest r2 ratio in each cluster using lowest rsq ratio is :")  
 subsetvar <- matrix(NA, nrow = length(index.list), ncol = 1)  
 for (i in min\_r2\_index[, "id"]) {  
 j <- min\_r2\_index[i, "min\_r2\_idx"]  
 i\_idx <- index.list[[i]][j]  
 subsetvar[i] <- as.character(groups.inputdf[groups.inputdf$ID == i\_idx,   
 ][, "var"])  
 }  
 # get columns for clusters which had only one variable  
 index.list.1.df <- as.data.frame(unlist(index.list.1))  
 colnames(index.list.1.df) <- c("col\_idx")  
 index.list.1.df$id <- seq.int(nrow(index.list.1.df))  
 subsetvarone <- matrix(NA, nrow = length(index.list.1), ncol = 1)  
 for (i in index.list.1.df[, "id"]) {  
 i\_idx <- index.list.1.df[i, "col\_idx"]  
 subsetvarone[i] <- as.character(groups.inputdf[groups.inputdf$ID ==   
 i\_idx, ][, "var"])  
 }  
 allsubvars <- rbind(subsetvar, subsetvarone)  
 return(allsubvars)  
}  
# Apply function r2clusterfun use cut = 10 as seen from pca around PC1 to  
# PC10 explian 90% of variance  
SchoolMain\_subset <- r2clusterfun(SchoolMain\_new\_numeric, insim = "spearman",   
 numcuttree = 10)



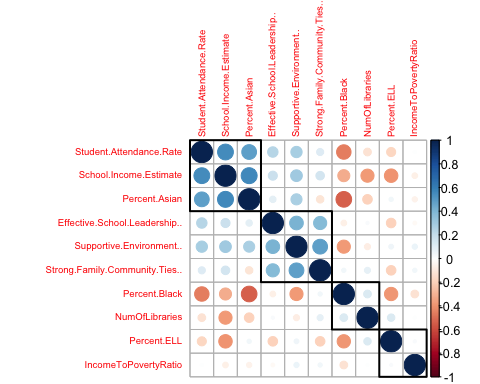
## [1] "The 10 variables with the lowest r2 ratio in each cluster using lowest rsq ratio is :"

# These are the predictor variables selected after numcuttree = 10 and min  
# rsq ratio selection.  
print(SchoolMain\_subset[, 1])  
## [1] "School.Income.Estimate" "Percent.ELL"   
## [3] "Percent.Asian" "Student.Attendance.Rate"   
## [5] "Supportive.Environment.." "Effective.School.Leadership.."   
## [7] "IncomeToPovertyRatio" "Percent.Black"   
## [9] "Strong.Family.Community.Ties.." "NumOfLibraries"

print("correlation between the above 10 elements is")  
## [1] "correlation between the above 10 elements is"

# Reduced dataset with only 10 variables is SchoolMain\_reduced  
SchoolMain\_reduced\_numeric <- SchoolMain\_new\_numeric[, c(SchoolMain\_subset)]  
SchoolMain\_subset\_corr <- round(cor(SchoolMain\_reduced\_numeric, method = c("spearman")),   
 2)  
# SchoolMain\_subset\_corr  
print("The subset of variables selected show no or very low correlation to each other.")  
## [1] "The subset of variables selected show no or very low correlation to each other."

# Check the data numeric volumns for correlation  
corrplot::corrplot(SchoolMain\_subset\_corr, order = "hclust", addrect = 4, method = "circle",   
 tl.cex = 0.6)



# Create a data set with reduced predictor variables with  
SchoolMain\_reduced <- cbind(SchoolMain\_reduced\_numeric, SchoolMain\_fctr\_resp)  
dim(SchoolMain\_reduced)  
## [1] 1217 12

# split the unreduced data into train and test  
set.seed(1000)  
train.school.index <- createDataPartition(y = SchoolMain\_reducedcorr$Average\_Proficiency,   
 p = 0.85, list = FALSE)  
trainschool <- SchoolMain\_reducedcorr[train.school.index, ]  
testschool <- SchoolMain\_reducedcorr[-train.school.index, ]  
dim(trainschool)  
## [1] 1036 20

dim(testschool)  
## [1] 181 20

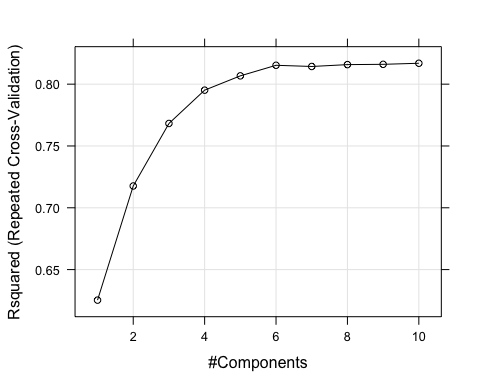
# split the reduced data into train and test  
set.seed(1000)  
train.reduced.school.index <- createDataPartition(y = SchoolMain\_reduced$Average\_Proficiency,   
 p = 0.85, list = FALSE)  
trainreducedschool <- SchoolMain\_reduced[train.reduced.school.index, ]  
testreducedschool <- SchoolMain\_reduced[-train.reduced.school.index, ]  
dim(trainreducedschool)  
## [1] 1036 12

dim(testreducedschool)  
## [1] 181 12

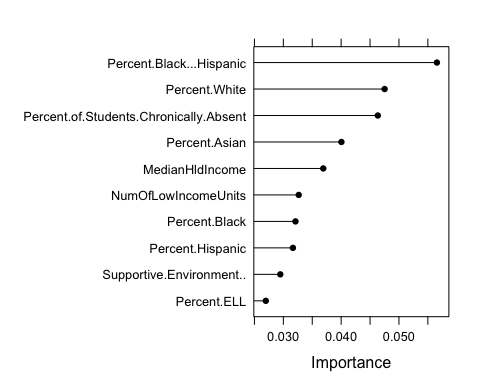
# Run the LM model  
set.seed(1000)  
lm\_school\_model = train(Average\_Proficiency ~ ., data = trainschool, method = "lm",   
 tuneLength = 10, preProcess = c("center", "scale"), trControl = trainControl(method = "repeatedcv",   
 number = 10))  
summary(lm\_school\_model)  
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.58334 -0.10876 -0.00368 0.08848 0.98583   
##   
## Coefficients:  
## Estimate Std. Error t value  
## (Intercept) 2.602268 0.005405 481.447  
## Percent.ELL -0.118541 0.007602 -15.594  
## Percent.Asian 0.122335 0.031455 3.889  
## Percent.Black -0.164532 0.032002 -5.141  
## Percent.Hispanic -0.100982 0.027217 -3.710  
## Percent.Black...Hispanic 0.086619 0.038409 2.255  
## Percent.White 0.089950 0.035017 2.569  
## Student.Attendance.Rate -0.187366 0.008429 -22.228  
## Percent.of.Students.Chronically.Absent -0.235048 0.010343 -22.726  
## Rigorous.Instruction.. 0.034395 0.007603 4.524  
## Supportive.Environment.. 0.044163 0.009231 4.784  
## Strong.Family.Community.Ties.. 0.015844 0.007785 2.035  
## Trust.. -0.032079 0.007761 -4.133  
## NumOfLibraries 0.010922 0.006938 1.574  
## NumOfLowIncomeUnits 0.023900 0.008519 2.806  
## MedianHldIncome -0.019261 0.008235 -2.339  
## ChildPoverty -0.031813 0.009767 -3.257  
## CountofSNAPHlds 0.047368 0.011136 4.254  
## HealthCoverage -0.014441 0.008684 -1.663  
## Community.School.1 -0.023572 0.005795 -4.067  
## Pr(>|t|)   
## (Intercept) < 2e-16 \*\*\*  
## Percent.ELL < 2e-16 \*\*\*  
## Percent.Asian 0.000107 \*\*\*  
## Percent.Black 3.27e-07 \*\*\*  
## Percent.Hispanic 0.000218 \*\*\*  
## Percent.Black...Hispanic 0.024334 \*   
## Percent.White 0.010347 \*   
## Student.Attendance.Rate < 2e-16 \*\*\*  
## Percent.of.Students.Chronically.Absent < 2e-16 \*\*\*  
## Rigorous.Instruction.. 6.78e-06 \*\*\*  
## Supportive.Environment.. 1.97e-06 \*\*\*  
## Strong.Family.Community.Ties.. 0.042087 \*   
## Trust.. 3.87e-05 \*\*\*  
## NumOfLibraries 0.115738   
## NumOfLowIncomeUnits 0.005119 \*\*   
## MedianHldIncome 0.019533 \*   
## ChildPoverty 0.001163 \*\*   
## CountofSNAPHlds 2.30e-05 \*\*\*  
## HealthCoverage 0.096636 .   
## Community.School.1 5.12e-05 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.174 on 1016 degrees of freedom  
## Multiple R-squared: 0.8238, Adjusted R-squared: 0.8205   
## F-statistic: 250 on 19 and 1016 DF, p-value: < 2.2e-16

# Run the LM model with pca  
set.seed(1000)  
lm\_pca\_school\_model = train(Average\_Proficiency ~ ., data = trainschool, method = "lm",   
 tuneLength = 10, preProcess = c("pca"), trControl = trainControl(method = "repeatedcv",   
 number = 10))  
summary(lm\_pca\_school\_model)  
##   
## Call:  
## lm(formula = .outcome ~ ., data = dat)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.54509 -0.13768 -0.02346 0.10392 1.11656   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.602e+00 6.626e-03 392.712 < 2e-16 \*\*\*  
## PC1 -1.376e-01 3.021e-03 -45.534 < 2e-16 \*\*\*  
## PC2 -3.749e-02 4.013e-03 -9.343 < 2e-16 \*\*\*  
## PC3 -7.814e-03 4.411e-03 -1.771 0.076789 .   
## PC4 5.461e-02 4.759e-03 11.475 < 2e-16 \*\*\*  
## PC5 3.075e-02 5.878e-03 5.231 2.04e-07 \*\*\*  
## PC6 -9.945e-02 6.319e-03 -15.738 < 2e-16 \*\*\*  
## PC7 3.990e-02 6.889e-03 5.792 9.28e-09 \*\*\*  
## PC8 -2.053e-03 7.042e-03 -0.292 0.770685   
## PC9 -7.636e-02 8.393e-03 -9.098 < 2e-16 \*\*\*  
## PC10 -7.494e-05 9.407e-03 -0.008 0.993645   
## PC11 -5.117e-02 1.008e-02 -5.076 4.57e-07 \*\*\*  
## PC12 1.067e-01 1.118e-02 9.545 < 2e-16 \*\*\*  
## PC13 4.043e-02 1.222e-02 3.309 0.000969 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.2133 on 1022 degrees of freedom  
## Multiple R-squared: 0.7336, Adjusted R-squared: 0.7302   
## F-statistic: 216.5 on 13 and 1022 DF, p-value: < 2.2e-16

set.seed(1000)  
# Run the PLS model  
pls\_school\_model = train(Average\_Proficiency ~ ., data = trainschool, method = "pls",   
 tuneLength = 10, preProcess = c("center", "scale"), trControl = trainControl(method = "repeatedcv",   
 number = 10))  
# summary(pls\_school\_model) head(pls\_school\_model$results)  
  
plot(pls\_school\_model, metric = "Rsquared")

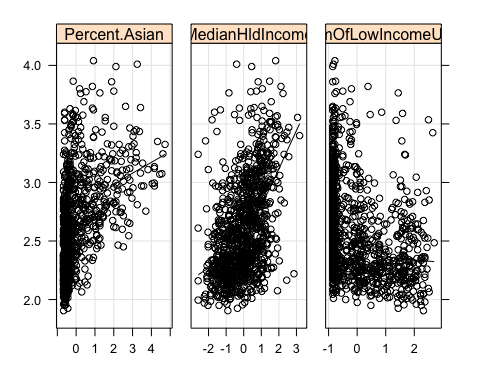
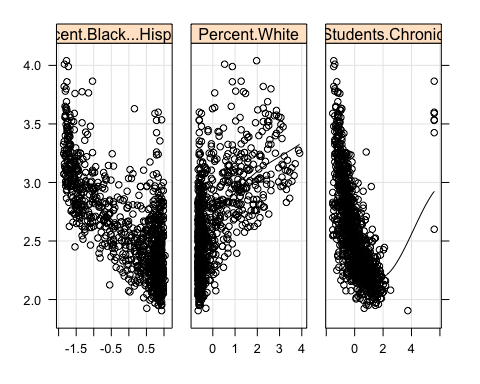


library(pls)  
pls\_school\_imp <- varImp(pls\_school\_model, scale = FALSE)  
plot(pls\_school\_imp, top = 10, scales = list(y = list(cex = 0.8)))



school\_order\_pls\_index <- order(abs(pls\_school\_imp$importance), decreasing = TRUE)  
top\_pls\_vars = rownames(pls\_school\_imp$importance)[school\_order\_pls\_index[c(1:6)]]  
top\_pls\_vars  
## [1] "Percent.Black...Hispanic"   
## [2] "Percent.White"   
## [3] "Percent.of.Students.Chronically.Absent"  
## [4] "Percent.Asian"   
## [5] "MedianHldIncome"   
## [6] "NumOfLowIncomeUnits"

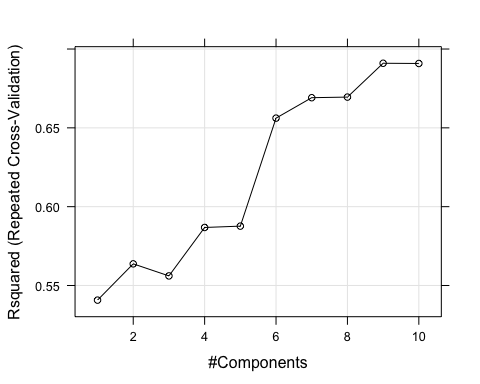
# Explore the univariate relationships of top 5 variables with Yield using  
# featureplot  
featurePlot(trainschool[, top\_pls\_vars], trainschool$Average\_Proficiency, plot = "scatter",   
 between = list(x = 1, y = 1), type = c("g", "p", "smooth"), layout = c(3,   
 1), labels = rep("", 2), warn = FALSE)



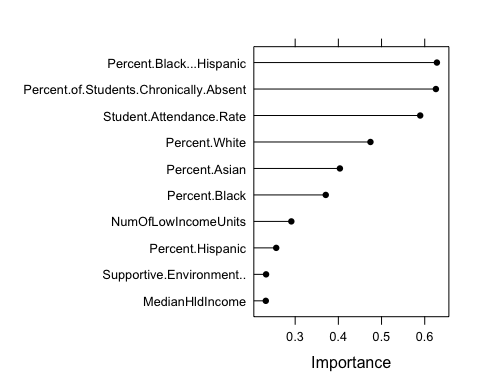
set.seed(1000)  
# Run the PCR model  
pcr\_school\_model = train(Average\_Proficiency ~ ., data = trainschool, method = "pcr",   
 tuneLength = 10, preProcess = c("center", "scale"), trControl = trainControl(method = "repeatedcv",   
 number = 10))  
summary(pcr\_school\_model)  
## Data: X dimension: 1036 19   
## Y dimension: 1036 1  
## Fit method: svdpc  
## Number of components considered: 9  
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps  
## X 25.35 39.71 51.6 61.81 68.51 74.3 79.17  
## .outcome 54.04 56.31 56.4 59.83 60.54 67.0 67.87  
## 8 comps 9 comps  
## X 83.84 87.12  
## .outcome 67.87 70.03

head(pcr\_school\_model$results)  
## ncomp RMSE Rsquared MAE RMSESD RsquaredSD MAESD  
## 1 1 0.2796475 0.5407185 0.2028526 0.03063004 0.10634277 0.01357237  
## 2 2 0.2736547 0.5637358 0.1968883 0.02793488 0.10512232 0.01743261  
## 3 3 0.2769644 0.5560777 0.1969548 0.03273748 0.11769496 0.01762724  
## 4 4 0.2656847 0.5868054 0.1889578 0.03142593 0.11077030 0.01792800  
## 5 5 0.2651279 0.5876561 0.1939877 0.02737210 0.10012889 0.01966341  
## 6 6 0.2412520 0.6561614 0.1761873 0.02347805 0.06936752 0.01766936

plot(pcr\_school\_model, metric = "Rsquared")

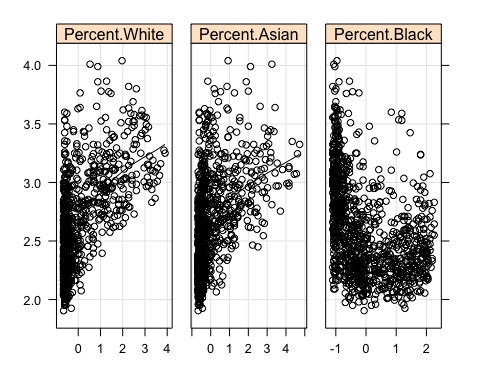
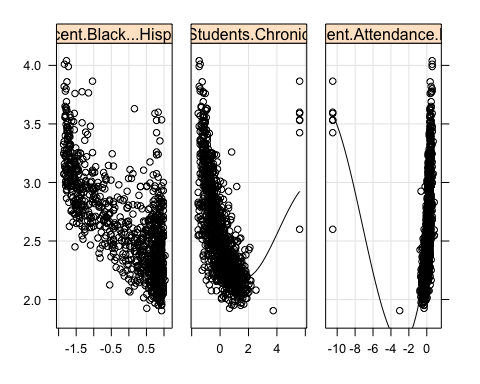


pcr\_school\_imp = varImp(pcr\_school\_model, scale = FALSE)  
plot(pcr\_school\_imp, top = 10, scales = list(y = list(cex = 0.8)))



school\_order\_pcr\_index <- order(abs(pcr\_school\_imp$importance), decreasing = TRUE)  
top\_pcr\_vars = rownames(pcr\_school\_imp$importance)[school\_order\_pcr\_index[c(1:6)]]  
top\_pcr\_vars  
## [1] "Percent.Black...Hispanic"   
## [2] "Percent.of.Students.Chronically.Absent"  
## [3] "Student.Attendance.Rate"   
## [4] "Percent.White"   
## [5] "Percent.Asian"   
## [6] "Percent.Black"

# Explore the univariate relationships of top 5 variables with Yield using  
# featureplot  
featurePlot(trainschool[, top\_pcr\_vars], trainschool$Average\_Proficiency, plot = "scatter",   
 between = list(x = 1, y = 1), type = c("g", "p", "smooth"), layout = c(3,   
 1), labels = rep("", 2), warn = FALSE)

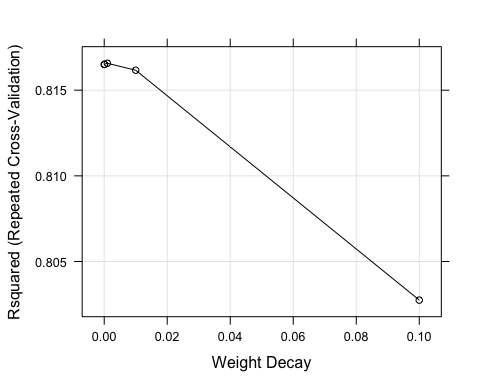


# Run the SVM model  
set.seed(1000)  
svm\_school\_model = train(Average\_Proficiency ~ ., data = trainschool, method = "svmLinear",   
 tuneLength = 10, trControl = trainControl(method = "repeatedcv", number = 10),   
 preProc = c("center", "scale"))  
# summary(svm\_school\_model)  
head(svm\_school\_model$results)  
## C RMSE Rsquared MAE RMSESD RsquaredSD MAESD  
## 1 1 0.1754597 0.8195157 0.1290315 0.01837664 0.03304322 0.01341512

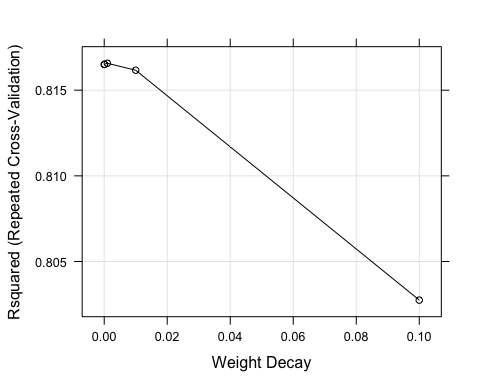
# Run the random forest model  
set.seed(1000)  
rf\_school\_model = train(Average\_Proficiency ~ ., data = trainschool, method = "rf",   
 tuneLength = 10, trControl = trainControl(method = "repeatedcv", number = 10),   
 preProc = c("center", "scale"))  
summary(rf\_school\_model)  
## Length Class Mode   
## call 4 -none- call   
## type 1 -none- character  
## predicted 1036 -none- numeric   
## mse 500 -none- numeric   
## rsq 500 -none- numeric   
## oob.times 1036 -none- numeric   
## importance 19 -none- numeric   
## importanceSD 0 -none- NULL   
## localImportance 0 -none- NULL   
## proximity 0 -none- NULL   
## ntree 1 -none- numeric   
## mtry 1 -none- numeric   
## forest 11 -none- list   
## coefs 0 -none- NULL   
## y 1036 -none- numeric   
## test 0 -none- NULL   
## inbag 0 -none- NULL   
## xNames 19 -none- character  
## problemType 1 -none- character  
## tuneValue 1 data.frame list   
## obsLevels 1 -none- logical   
## param 0 -none- list

head(rf\_school\_model$results)  
## mtry RMSE Rsquared MAE RMSESD RsquaredSD MAESD  
## 1 2 0.1835608 0.8095324 0.1377115 0.02192465 0.03889271 0.01280627  
## 2 3 0.1776427 0.8187236 0.1339499 0.02114752 0.03747631 0.01244852  
## 3 5 0.1743759 0.8230498 0.1325421 0.02007149 0.03464191 0.01213274  
## 4 7 0.1737129 0.8237009 0.1321444 0.01927438 0.03372099 0.01162485  
## 5 9 0.1736811 0.8235716 0.1320067 0.01899197 0.03242861 0.01149992  
## 6 11 0.1741404 0.8223065 0.1322510 0.01772293 0.03050220 0.01099485

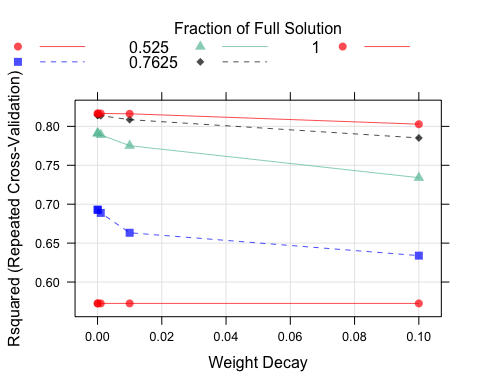
# Ridge Regression Model  
set.seed(1000)  
ridge\_school\_model <- train(Average\_Proficiency ~ ., data = trainschool, method = "ridge",   
 trControl = trainControl(method = "repeatedcv", repeats = 10), preProc = c("center",   
 "scale"), tuneLength = 5)  
# summary(ridge\_school\_model)  
  
plot(ridge\_school\_model, metric = "Rsquared")



# LASSO Model  
set.seed(1000)  
lasso\_school\_model <- train(Average\_Proficiency ~ ., data = trainschool, method = "lasso",   
 trControl = trainControl(method = "repeatedcv", repeats = 10), preProc = c("center",   
 "scale"), tuneLength = 5)  
# summary(lasso\_school\_model)  
plot(ridge\_school\_model, metric = "Rsquared")



# ENET Model (ELastic net Regression)  
set.seed(1000)  
library(elasticnet)  
enet\_school\_model <- train(Average\_Proficiency ~ ., data = trainschool, method = "enet",   
 trControl = trainControl(method = "repeatedcv", repeats = 10), preProc = c("center",   
 "scale"), tuneLength = 5)  
# summary(enet\_school\_model)  
plot(enet\_school\_model, metric = "Rsquared")



# Compare the model, this works only when trcontrol or sampling method is  
# same in all the models used.  
resamp\_school\_1 = resamples(list(lm = lm\_school\_model, lm.pca = lm\_pca\_school\_model,   
 pcr = pcr\_school\_model, rf = rf\_school\_model, svm = svm\_school\_model, pls = pls\_school\_model))  
print(summary(resamp\_school\_1))  
##   
## Call:  
## summary.resamples(object = resamp\_school\_1)  
##   
## Models: lm, lm.pca, pcr, rf, svm, pls   
## Number of resamples: 10   
##   
## MAE   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## lm 0.1070685 0.1180860 0.1366689 0.1300807 0.1397910 0.1467754 0  
## lm.pca 0.1357460 0.1475072 0.1668308 0.1618575 0.1725009 0.1894222 0  
## pcr 0.1485546 0.1594144 0.1697648 0.1716794 0.1815798 0.2044287 0  
## rf 0.1048389 0.1287288 0.1336964 0.1320067 0.1376371 0.1448903 0  
## svm 0.1079733 0.1176457 0.1353986 0.1290315 0.1375384 0.1457756 0  
## pls 0.1085478 0.1188492 0.1372913 0.1314325 0.1411022 0.1497866 0  
##   
## RMSE   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## lm 0.1537757 0.1591449 0.1742008 0.1760032 0.1866990 0.2061216 0  
## lm.pca 0.1891725 0.2000402 0.2180711 0.2174238 0.2313447 0.2559584 0  
## pcr 0.2067503 0.2147798 0.2275585 0.2287004 0.2325577 0.2738308 0  
## rf 0.1302318 0.1672890 0.1775464 0.1736811 0.1870571 0.1924666 0  
## svm 0.1503215 0.1606478 0.1746747 0.1754597 0.1853086 0.2063181 0  
## pls 0.1541509 0.1612387 0.1742424 0.1766730 0.1867431 0.2063027 0  
##   
## Rsquared   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## lm 0.7541504 0.7990137 0.8252071 0.8182619 0.8304570 0.8682100 0  
## lm.pca 0.6165553 0.7091109 0.7330969 0.7219755 0.7491999 0.7859075 0  
## pcr 0.5632684 0.6765649 0.7017807 0.6909685 0.7267588 0.7463176 0  
## rf 0.7716673 0.8022952 0.8246424 0.8235716 0.8451246 0.8796140 0  
## svm 0.7551649 0.7995862 0.8280184 0.8195157 0.8337569 0.8638520 0  
## pls 0.7537791 0.7996061 0.8233674 0.8169100 0.8283084 0.8677423 0

resamp\_school = resamples(list(ridge = ridge\_school\_model, lasso = lasso\_school\_model,   
 enet = enet\_school\_model))  
print(summary(resamp\_school))  
##   
## Call:  
## summary.resamples(object = resamp\_school)  
##   
## Models: ridge, lasso, enet   
## Number of resamples: 100   
##   
## MAE   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## ridge 0.09890014 0.1237031 0.1305738 0.1303046 0.1382988 0.1556626 0  
## lasso 0.09843330 0.1242175 0.1310940 0.1305716 0.1387580 0.1560242 0  
## enet 0.09890014 0.1237031 0.1305738 0.1303046 0.1382988 0.1556626 0  
##   
## RMSE   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## ridge 0.1405019 0.1644976 0.1742045 0.1761714 0.1886771 0.2198966 0  
## lasso 0.1401326 0.1641746 0.1743083 0.1763594 0.1892566 0.2202151 0  
## enet 0.1405019 0.1644976 0.1742045 0.1761714 0.1886771 0.2198966 0  
##   
## Rsquared   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## ridge 0.6922914 0.7872156 0.8211979 0.8165708 0.8475693 0.8902503 0  
## lasso 0.6913928 0.7873846 0.8213492 0.8161379 0.8468088 0.8893012 0  
## enet 0.6922914 0.7872156 0.8211979 0.8165708 0.8475693 0.8902503 0

print("RMSE an R2 for train data for these models:-")  
## [1] "RMSE an R2 for train data for these models:-"

Average\_Proficiency\_lm\_hat = predict(lm\_school\_model, newdata = subset(trainschool,   
 select = -c(Average\_Proficiency)))  
r2\_lm = cor(Average\_Proficiency\_lm\_hat, trainschool$Average\_Proficiency, method = "pearson")^2  
rmse\_lm = sqrt(mean((Average\_Proficiency\_lm\_hat - trainschool$Average\_Proficiency)^2))  
  
Average\_Proficiency\_pls\_hat = predict(pls\_school\_model, newdata = subset(trainschool,   
 select = -c(Average\_Proficiency)))  
r2\_pls = cor(Average\_Proficiency\_pls\_hat, trainschool$Average\_Proficiency, method = "pearson")^2  
rmse\_pls = sqrt(mean((Average\_Proficiency\_pls\_hat - trainschool$Average\_Proficiency)^2))  
  
Average\_Proficiency\_pcr\_hat = predict(pcr\_school\_model, newdata = subset(trainschool,   
 select = -c(Average\_Proficiency)))  
r2\_pcr = cor(Average\_Proficiency\_pcr\_hat, trainschool$Average\_Proficiency, method = "pearson")^2  
rmse\_pcr = sqrt(mean((Average\_Proficiency\_pcr\_hat - trainschool$Average\_Proficiency)^2))  
  
Average\_Proficiency\_rf\_hat = predict(rf\_school\_model, newdata = subset(trainschool,   
 select = -c(Average\_Proficiency)))  
r2\_rf = cor(Average\_Proficiency\_rf\_hat, trainschool$Average\_Proficiency, method = "pearson")^2  
rmse\_rf = sqrt(mean((Average\_Proficiency\_rf\_hat - trainschool$Average\_Proficiency)^2))  
  
Average\_Proficiency\_svm\_hat = predict(svm\_school\_model, newdata = subset(trainschool,   
 select = -c(Average\_Proficiency)))  
r2\_svm = cor(Average\_Proficiency\_svm\_hat, trainschool$Average\_Proficiency, method = "pearson")^2  
rmse\_svm = sqrt(mean((Average\_Proficiency\_svm\_hat - trainschool$Average\_Proficiency)^2))  
  
Average\_Proficiency\_ridge\_hat = predict(ridge\_school\_model, newdata = subset(trainschool,   
 select = -c(Average\_Proficiency)))  
r2\_ridge = cor(Average\_Proficiency\_ridge\_hat, trainschool$Average\_Proficiency,   
 method = "pearson")^2  
rmse\_ridge = sqrt(mean((Average\_Proficiency\_ridge\_hat - trainschool$Average\_Proficiency)^2))  
  
Average\_Proficiency\_lasso\_hat = predict(lasso\_school\_model, newdata = subset(trainschool,   
 select = -c(Average\_Proficiency)))  
r2\_lasso = cor(Average\_Proficiency\_lasso\_hat, trainschool$Average\_Proficiency,   
 method = "pearson")^2  
rmse\_lasso = sqrt(mean((Average\_Proficiency\_lasso\_hat - trainschool$Average\_Proficiency)^2))  
  
train.rmse.table <- rbind(rmse\_lm, rmse\_pls, rmse\_pcr, rmse\_rf, rmse\_svm, rmse\_ridge,   
 rmse\_lasso)  
train.rmse.table  
## [,1]  
## rmse\_lm 0.17228604  
## rmse\_pls 0.17328262  
## rmse\_pcr 0.22469489  
## rmse\_rf 0.07225964  
## rmse\_svm 0.17377584  
## rmse\_ridge 0.17229341  
## rmse\_lasso 0.17247428

train.r2.table <- rbind(r2\_lm, r2\_pls, r2\_pcr, r2\_rf, r2\_svm, r2\_ridge, r2\_lasso)  
train.r2.table  
## [,1]  
## r2\_lm 0.8238073  
## r2\_pls 0.8217631  
## r2\_pcr 0.7003087  
## r2\_rf 0.9721815  
## r2\_svm 0.8212481  
## r2\_ridge 0.8237923  
## r2\_lasso 0.8234368

print("RMSE an R2 for test data for these models:-")  
## [1] "RMSE an R2 for test data for these models:-"

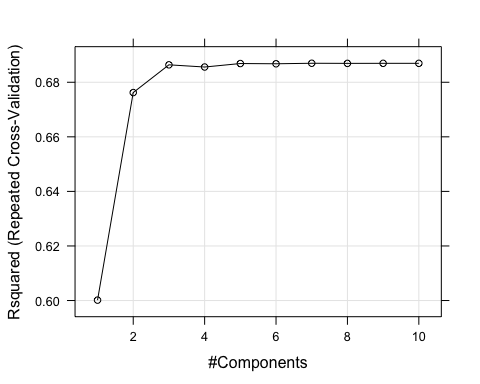
Average\_Proficiency\_lm\_hat = predict(lm\_school\_model, newdata = subset(testschool,   
 select = -c(Average\_Proficiency)))  
r2\_lm = cor(Average\_Proficiency\_lm\_hat, testschool$Average\_Proficiency, method = "pearson")^2  
rmse\_lm = sqrt(mean((Average\_Proficiency\_lm\_hat - testschool$Average\_Proficiency)^2))  
  
Average\_Proficiency\_pls\_hat = predict(pls\_school\_model, newdata = subset(testschool,   
 select = -c(Average\_Proficiency)))  
r2\_pls = cor(Average\_Proficiency\_pls\_hat, testschool$Average\_Proficiency, method = "pearson")^2  
rmse\_pls = sqrt(mean((Average\_Proficiency\_pls\_hat - testschool$Average\_Proficiency)^2))  
  
Average\_Proficiency\_pcr\_hat = predict(pcr\_school\_model, newdata = subset(testschool,   
 select = -c(Average\_Proficiency)))  
r2\_pcr = cor(Average\_Proficiency\_pcr\_hat, testschool$Average\_Proficiency, method = "pearson")^2  
rmse\_pcr = sqrt(mean((Average\_Proficiency\_pcr\_hat - testschool$Average\_Proficiency)^2))  
  
Average\_Proficiency\_rf\_hat = predict(rf\_school\_model, newdata = subset(testschool,   
 select = -c(Average\_Proficiency)))  
r2\_rf = cor(Average\_Proficiency\_rf\_hat, testschool$Average\_Proficiency, method = "pearson")^2  
rmse\_rf = sqrt(mean((Average\_Proficiency\_rf\_hat - testschool$Average\_Proficiency)^2))  
  
Average\_Proficiency\_svm\_hat = predict(svm\_school\_model, newdata = subset(testschool,   
 select = -c(Average\_Proficiency)))  
r2\_svm = cor(Average\_Proficiency\_svm\_hat, testschool$Average\_Proficiency, method = "pearson")^2  
rmse\_svm = sqrt(mean((Average\_Proficiency\_svm\_hat - testschool$Average\_Proficiency)^2))  
  
Average\_Proficiency\_ridge\_hat = predict(ridge\_school\_model, newdata = subset(testschool,   
 select = -c(Average\_Proficiency)))  
r2\_ridge = cor(Average\_Proficiency\_ridge\_hat, testschool$Average\_Proficiency,   
 method = "pearson")^2  
rmse\_ridge = sqrt(mean((Average\_Proficiency\_ridge\_hat - testschool$Average\_Proficiency)^2))  
  
Average\_Proficiency\_lasso\_hat = predict(lasso\_school\_model, newdata = subset(testschool,   
 select = -c(Average\_Proficiency)))  
r2\_lasso = cor(Average\_Proficiency\_lasso\_hat, testschool$Average\_Proficiency,   
 method = "pearson")^2  
rmse\_lasso = sqrt(mean((Average\_Proficiency\_lasso\_hat - testschool$Average\_Proficiency)^2))  
  
test.rmse.table <- rbind(rmse\_lm, rmse\_pls, rmse\_pcr, rmse\_rf, rmse\_svm, rmse\_ridge,   
 rmse\_lasso)  
test.rmse.table  
## [,1]  
## rmse\_lm 0.1890190  
## rmse\_pls 0.1887924  
## rmse\_pcr 0.2613816  
## rmse\_rf 0.1904957  
## rmse\_svm 0.1901785  
## rmse\_ridge 0.1889537  
## rmse\_lasso 0.1882144

test.r2.table <- rbind(r2\_lm, r2\_pls, r2\_pcr, r2\_rf, r2\_svm, r2\_ridge, r2\_lasso)  
test.r2.table  
## [,1]  
## r2\_lm 0.7873265  
## r2\_pls 0.7879323  
## r2\_pcr 0.5920728  
## r2\_rf 0.7847317  
## r2\_svm 0.7860609  
## r2\_ridge 0.7874544  
## r2\_lasso 0.7889797

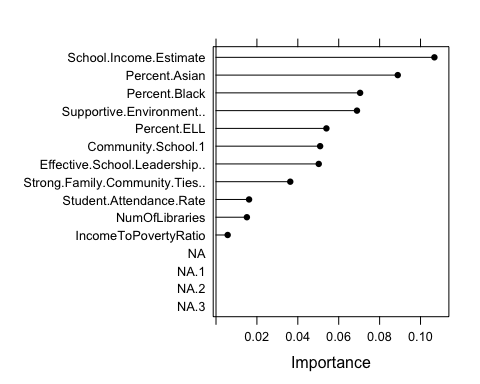
# Run Models on reduced 10 variables after rsq ratio based reduction dataset  
# Run the PLS model  
set.seed(10)  
pls\_school\_reduced\_model = train(Average\_Proficiency ~ ., data = trainreducedschool,   
 method = "pls", tuneLength = 10, preProcess = c("center", "scale"), trControl = trainControl(method = "repeatedcv",   
 number = 10))  
summary(pls\_school\_reduced\_model)  
## Data: X dimension: 1036 11   
## Y dimension: 1036 1  
## Fit method: oscorespls  
## Number of components considered: 7  
## TRAINING: % variance explained  
## 1 comps 2 comps 3 comps 4 comps 5 comps 6 comps 7 comps  
## X 21.02 30.33 41.60 49.66 61.43 69.22 74.46  
## .outcome 59.46 67.77 68.46 68.71 68.77 68.78 68.78

head(pls\_school\_reduced\_model$results)  
## ncomp RMSE Rsquared MAE RMSESD RsquaredSD MAESD  
## 1 1 0.2615320 0.6001462 0.1959474 0.05461148 0.11228502 0.03644581  
## 2 2 0.2352148 0.6762246 0.1804936 0.03969794 0.07670236 0.02809049  
## 3 3 0.2312942 0.6863925 0.1774939 0.03971397 0.07552843 0.02710335  
## 4 4 0.2315310 0.6855858 0.1766251 0.04021071 0.07676126 0.02670676  
## 5 5 0.2311490 0.6868625 0.1769791 0.03931486 0.07458828 0.02624344  
## 6 6 0.2312108 0.6867748 0.1770614 0.03955089 0.07503762 0.02620907

plot(pls\_school\_reduced\_model, metric = "Rsquared")

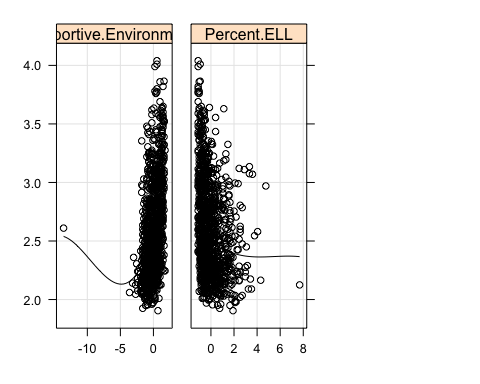
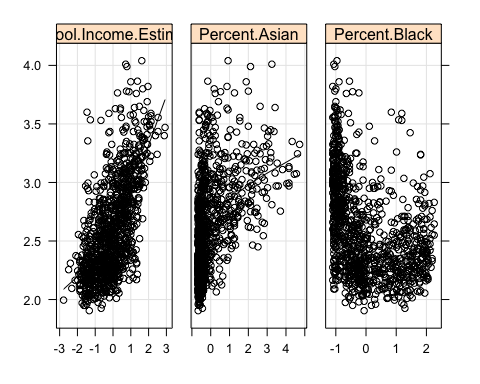


library(pls)  
pls\_school\_reduced\_imp <- varImp(pls\_school\_reduced\_model, scale = FALSE)  
plot(pls\_school\_reduced\_imp, top = 15, scales = list(y = list(cex = 0.8)))



reduced\_school\_order\_pls\_index <- order(abs(pls\_school\_reduced\_imp$importance),   
 decreasing = TRUE)  
reduced\_top\_pls\_vars = rownames(pls\_school\_reduced\_imp$importance)[reduced\_school\_order\_pls\_index[c(1:5)]]  
reduced\_top\_pls\_vars  
## [1] "School.Income.Estimate" "Percent.Asian"   
## [3] "Percent.Black" "Supportive.Environment.."  
## [5] "Percent.ELL"

# Explore the univariate relationships of top 5 variables with Yield using  
# featureplot  
featurePlot(trainreducedschool[, reduced\_top\_pls\_vars], trainreducedschool$Average\_Proficiency,   
 plot = "scatter", between = list(x = 1, y = 1), type = c("g", "p", "smooth"),   
 layout = c(3, 1), labels = rep("", 2), warn = FALSE)



Average\_Proficiency\_pls\_reduced\_hat = predict(pls\_school\_reduced\_model, newdata = subset(trainreducedschool,   
 select = -c(Average\_Proficiency)))  
r2\_train\_pls = cor(Average\_Proficiency\_pls\_reduced\_hat, trainreducedschool$Average\_Proficiency,   
 method = "pearson")^2  
r2\_train\_pls  
## [1] 0.6877945

rmse\_train\_pls = sqrt(mean((Average\_Proficiency\_pls\_reduced\_hat - trainreducedschool$Average\_Proficiency)^2))  
rmse\_train\_pls  
## [1] 0.2293382

Average\_Proficiency\_pls\_reduced\_hat = predict(pls\_school\_reduced\_model, newdata = subset(testreducedschool,   
 select = -c(Average\_Proficiency)))  
r2\_test\_pls = cor(Average\_Proficiency\_pls\_reduced\_hat, testreducedschool$Average\_Proficiency,   
 method = "pearson")^2  
r2\_test\_pls  
## [1] 0.6278892

rmse\_test\_pls = sqrt(mean((Average\_Proficiency\_pls\_reduced\_hat - testreducedschool$Average\_Proficiency)^2))  
rmse\_test\_pls  
## [1] 0.2495613